REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** 26.Oct.98	**3. REPORT TYPE AND DATES COVERED** MAJOR REPORT

4. TITLE AND SUBTITLE DEVELOPMENT OF A STAR IDENTIFICATION ALGORITHM	**5. FUNDING NUMBERS**
6. AUTHOR(S) 2D LT SHUMATE JON L	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) UNIVERSITY OF COLORADO AT COLORADO SPRINGS	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) THE DEPARTMENT OF THE AIR FORCE AFIT/CIA, BLDG 125 2950 P STREET WPAFB OH 45433	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** 98-019
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION AVAILABILITY STATEMENT Unlimited distribution In Accordance With AFI 35-205/AFIT Sup 1	**12b. DISTRIBUTION CODE**
13. ABSTRACT *(Maximum 200 words)*	
14. SUBJECT TERMS	**15. NUMBER OF PAGES**
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**

Development of a Star Identification Algorithm

Jon L. Shumate

04 May 1998

Master of Engineering in Space Operations Degree

Abstract

This paper is an overview of some of the current technology in star identification, and an explanation of an algorithm implemented to attempt to identify single stars. The overview section of the paper will describe the current trend in star imaging hardware, and how the charge coupled device (CCD) has revolutionized the star imaging process (No in-depth description of CCD technology will be made). It will also briefly describe the identification algorithms used prior to the advent of CCD technology. A portion of the paper will describe the triplet star identification algorithms, and the bright star matching technique. The bulk of the paper will be a detailed description of the single-star identification algorithm developed, an explanation of its implementation into MATHCAD format, and some explanation of problems associated with the implementation. An additional section overviews how well the algorithm works, as tested for a variety of stars. The conclusion of the paper will include a basic assessment of the value of the algorithm, and possibilities for future improvements to the algorithm.

Table Of Contents

Background

Star trackers have historically been a superior means of attitude determination. Because the celestial sphere remains essentially fixed, star positions provide an excellent inertial reference frame for determining attitude. Other means of attitude determination, such magnetometer measurements, sun sensing, or inertial measurement do not generate anything near the accuracy obtained by systems which use the celestial sphere stars as a reference [3].

The basic idea behind a star-tracking type system is as follows: given a satellite's position in its orbit, and knowledge of a vector from a satellite coordinate system to a given star, the attitude of a satellite relative to the Earth can be determined. This attitude can then be altered by attitude control systems such as reaction wheels, for pointing purposes. Accurate pointing capabilities are extremely important for remote sensing as well as astronomy missions, in which the pointing accuracy requirements for a mission are usually less than a degree [3].

In order to obtain the determination of attitude necessary for generating the pointing capabilities mentioned above, it is first necessary to generate an accurate star vector from the satellite in question to a star in the celestial sphere. In order to generate an accurate star vector, it is necessary to have an idea of which star, or which point in the celestial sphere the star tracker is oriented towards. This star identification process is one of the principle tasks of any on-board star tracker, and increasing the speed at which star identification is achieved will increase the speed of attitude control for fine guidance purposes.

From a hardware standpoint, initial efforts towards star identification involved the use of slit star sensors, or image dissector tube fixed head star trackers. Slit sensors made observations of star magnitude and position based on light intensity patterns crossing slits in the instrument. By timing the duration required for a star to cross a slit (called transit time[4]), it is possible to generate star positions in a given angular frame. Image dissector tubes measure the position of a star on a focal plane, based on planar geometry. After observations are made using slit sensors or image dissector tubes, the stars in the field of view are identified, by comparing the observed stars to stars in an on-board catalog.

While these older types of hardware worked, identification methods for these systems usually required some a-priori knowledge of the spacecraft attitude. Basic algorithms for identifying specific stars in these older systems were based on the "direct match" technique, which matches observations of star fields to catalog stars, with the catalog star being within a certain tolerance of a given position in the field of view. The "angular separation" technique was based on measuring the angular distances between stars in an observation, and then comparing those angular measurements to a catalog of angles for specific stars. The "phase matching" technique determines a phase angle around the spin axis of a spinning spacecraft, by matching star observations about the spin axis to a catalog of star positions for the given spin axis. It basically determines a phase angle by comparing the observed star field to a catalog of possible observations for a given spin axis, and then determining what possible angles at which the observation could be made, based on the observed stars. Finally, the "discrete attitude variation" technique, was used to determine attitude without prior information, and was based on pointing the spacecraft in various directions, and then using one of the previously mentioned methods to generate a "score" as to how precisely that attitude has been determined. After repeated observations, attitude can be determined, although it may take many minutes to do this [4].

The new technology in star tracking is the charge coupled device, or CCD [6]. Instruments using a CCD provide an actual image of a star field, with the number of pixels in the image being determined by the size and quality of the CCD itself. Each illuminated pixel in the image will have an intensity associated with it, based on the amount of light striking the portion of the CCD corresponding to the pixel. Large values for the numbers associated with each pixel represent brightness, whereas small values for the given numbers represent lack of brightness. The amount of brightness is a function of the amount of light striking the CCD, and hence the brightness of the star that is being imaged by a particular portion of the CCD. The output of the CCD will look like a matrix of numbers, low numbers corresponding to dim regions in the field of view, and `high numbers corresponding to bright regions within the field of view, such as stars.

Star trackers using CCDs have the advantage of being able to determine the visual magnitude of stars, and track multiple stars. CCDs are a substantial improvement from the fixed head star trackers, which could only track one star at a time, and were incapable of making high-quality magnitude comparisons [6]. CCDs also have the advantage of being relatively light-weight, although they do present some challenges in terms of temperature control and calibration.

Current CCD identification techniques have centered primarily around the use of the distances between stars, and angular measurements between stars-- these angles being taken about a single star to be identified. Data concerning the angular displacement in arc-seconds of star pairs relative to an individual star is cataloged, as are the distances in arc-seconds between stars. When an observation is made of a star field by a CCD device, the angular data and/or distance data is compared to the data for all of the stars in the catalog. The data set from the catalog star which most closely matches the data set for the observed star is found, and the observed star can be identified as that specific catalog star [10]. CCD techniques can be used to determine attitude without any a priori information, due to the high accuracy of CCD measurements, and the speed of modern star identification algorithms and processing.

Introduction

Problem statement:

The problem addressed here will be star identification. Given a star field generated from a CCD observation, it is necessary to identify a star in the field of view. This is done based on comparing observed features of the star field with data for specific stars, kept in a catalog. It is assumed that the field can be generated in a random orientation, meaning that the same field when observed twice may be rotated relative to other observations of the same field. This implies that the identification algorithm will need to be rotationally independent. Further assumptions for the problem will be described for different solution techniques.

Problem relevance:

Star identification is an important part of the attitude acquisition process. Because these instruments are relatively light-weight, they may eventually replace other means of attitude determination aboard spacecraft. In order for these instruments to meet the sometimes rapid pointing requirements of modern space missions, it is becoming more and more important for attitude to be determined rapidly. Improving star identification algorithms can speed up attitude determination, making this type of research very beneficial to the attitude determination process.

Description of Contents:

This paper addresses the problem of star identification, and the algorithms used in actually identifying stars. Obviously, there are many different methods used in attempting to identify stars, and two of those will be presented here. In addition to this, an attempt has been made to develop and implement a new identification algorithm, which is actually somewhat similar to one of the algorithms presented here.

The remainder of this paper is organized as follows: The concepts of two of the more common CCD star identification techniques will be presented. A somewhat new approach to the problem will then be described. The assumptions made in the detailed algorithm designed for that approach will then be listed. The detailed algorithm and its implementation are included, as are some discussions of the limitations and applications of these types of algorithms. Concluding remarks will be directed towards the effectiveness of the new algorithm developed, and some lessons learned for future attempts to solve this problem.

Tri-Star Techniques

The basic idea behind what is referred to as "tri-star" matching techniques is the observation of the angles, as well as the distances between observed stars, and comparison of that data to data in a star catalog. Observed stars are grouped into "triplets," a triplet being a group of three observed stars, as shown in Figure 1:

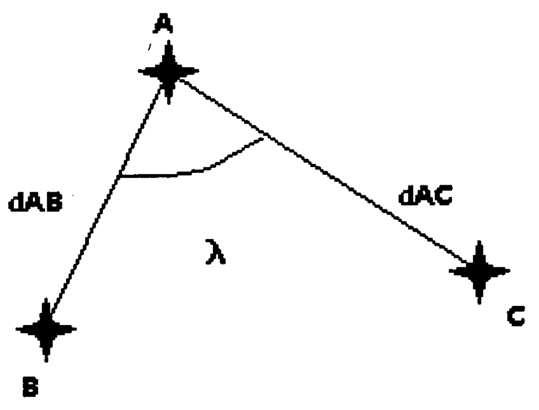


Figure 1: Star triplet

In the above triplet, the star being identified is star A. It forms a triplet with stars B and C. The distances dAB and dAC are the distances between stars A and B and stars A and C [7]. Common star identification algorithms oriented towards the use of triplets will use the angle λ, as shown above, as an initial identifying feature of a star. The angle λ is taken between the star of interest, and the two nearest stars. In conjunction with the distances dAB and dAC this angle is placed into a data set. This data set would probably look something like (λ,dAB,dAC). This triplet data would then be compared against a catalog of triplets for stars within the observed magnitude range, until a matching star triplet within the catalog was found.

Initially this technique may look rather easy to implement, but actual catalogs can become somewhat ambiguous, including many triplet data sets for each individual star. The reason for this is the fact that there is usually a certain amount of error in any observation, so multiple data sets are included in the catalog in hopes of generating a match. In addition to this, there is often a difficulty in CCD observations measuring different stars as the B and C points of the above-mentioned triplet. This might happen if a dimmer star near the star being identified were registered as a star in the field of view, changing the size and shape of the observed triplet for the star being identified. To compensate for this, the data sets in the catalog include multiple distances and angular measurements. Star catalogs can have up to 150 triplet sets for each individual star [6]. In actual application, every star in a field of view will have its triplet calculated, and compared to the stars in an onboard catalog. Attitude determination is then done by comparing all of the matches to positions of those matches on the celestial sphere. Lindsey [6] states that a certain degree of "common sense" logic is used in algorithms determining what star field is being observed. If most of the triplet matches in a field correspond to one portion of the celestial sphere, and a few are in different regions of the sky, the region with the most matches is obviously the region being observed by a star tracker.

Matched Group Techniques

Matched group techniques are another form of star identification, based on magnitude matches of multiple stars. In this technique anywhere from 8 to 12 stars are used in the matching algorithm [9]. Figure 2 illustrates the basic setup:

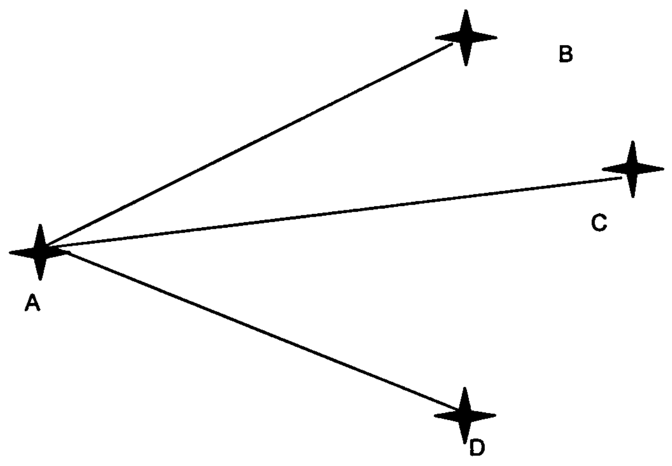


Figure 2: Match Group technique

In the above diagram, star A is a kernel star, meaning that it will form star pairs with all of the other stars in the given match group (Usually limited to a maximum of 8 to 12 stars [9]). Together, stars A, B, C, and D form a match group. The generation of this match group occurred because of matching of star pairs. Pair matching involves finding the visual magnitudes of observed stars, and then attempting to match pairs of observed star magnitudes to pairs of stars of the same magnitudes kept in a star catalog. In the above example, the kernel star A is matched to stars B, C, and D, forming three star pairs. These pairs are compared to pairs in a catalog, to determine which pairs they might be. The pairs AB, AC, and AD form a match group, the quality of which is determined by the differences between the observed pairs (AB, AC, and AD) and the catalog pairs they are most similar to. The general search algorithm will find numerous match groups in a field of view. The match group of highest quality is selected as a correct match group, representative of the catalog stars in the match group, and the identification is complete. The match group technique is probably superior to the tri-star technique. The reason for this is that the catalog required for a match group algorithm is an order of magnitude smaller than that required for a tri-star algorithm [9].

Modified Tri-Star Approach

The approach toward star identification taken in this paper has been oriented towards the format used in the tri-star algorithm. The difference is in the number of stars chosen for comparison. While the tri-star approach is oriented towards identification of one star based on two stars located near it, a multi-star approach seeks to identify the star in the center of a field of view based on observations of all of the stars around it. A diagram of this approach is shown in figure 3:

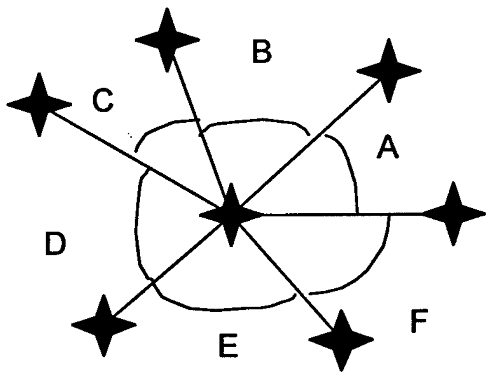


Figure 3: Modified tri-star approach

Once an observation has been made, the angles between all of the stars surrounding the central star are calculated. In the above figure, these angles would be A, B, C, D, E, and F respectively. These angles are then compared to angles defined for a specific star, kept in a star catalog. To avoid any discrepancies in the order in which the angles are compared, they are arranged in order from lowest to highest. The catalog star with the fewest differences in angle size and number from an observed star will be treated as the star being observed. This approach does not take into account the distance information from stars to the central star, but it is unclear as to whether or not adding that information would assist in the identification process.

Algorithm and Implementation

The implementation of a multi-angle identification algorithm was done in MATHCAD, a mathematical software package useful for numerical calculations, and capable of basic programming functions (The complete implementation is shown in Appendix A). The algorithm was designed to find the best match for a single star out of a catalog of 10 different stars. The data input into the program is a set bitmap files taken from the digitized sky survey. The files are 5 minutes by 5 minutes in angular size, and form an array of 176 X 176 pixel elements. These files are taken from the Digitized Sky Survey produced by the Space Telescope Science Institute, and are a decent representation of what a CCD instrument will generate after some basic image processing (They are actually digitized photographic plate images). The algorithm assumes that the data arrays placed into it have already been processed, and are of reasonably high resolution, meaning that it is possible to distinguish stars in the field of view. The output of the algorithm is a data string, which will consist of the angular separations of the stars surrounding the central star in the field of view, and a number corresponding to the catalog star which the observed star is most similar to.

Assumptions

In order for the multi-star algorithm to be implemented correctly, there are some basic assumptions that are made about the data being placed into it. One of the first assumptions is that the field of view for any observed star is the same as the field of view used to generate the star catalog. This is important because in the event that the star catalog field of view is larger or smaller than the field of view used for observation, there could be a large difference between the number of angles found for a catalog star and the number of angles that could be observed in the observational field of view. For instance, in Figures 4 and 5, different fields of view are applied to the same star. Notice that the largest field of view is capable of observing 5 stars around the central star, whereas the smaller field of view is only capable of observing 2 stars. Although the same star is being observed, if Figure 4 were the catalog star, and Figure 5 were the observed star, or vice versa, the star would not be properly matched.

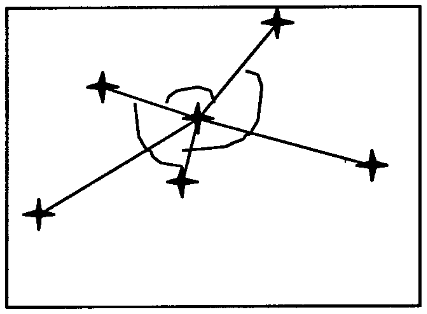


Figure 4.

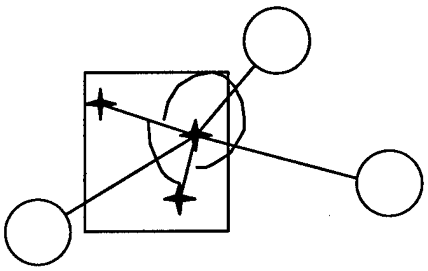


Figure 5.

There is another assumption made in reference to the field of view. Since most star trackers have a relatively wide field of view (on the order of degrees), and many stars can occupy a wide field of view, the problem here will be limited to a field of view much smaller than that of the normal star tracker. Part of the reason for this is that the data used for implementation of the star identification algorithm presented here was taken from the Digitized Sky Survey (DSS), which represents digitization of photographic plate images of the entire sky, taken with an enormous telescope on top of Mt. Palomar, in California. Since the images were taken with such a high quality instrument, there will be considerably more stars found in the field of view of the DSS then would be detected in the field of view of an ordinary star tracker, which does not have the optics or exposure time which was used on Palomar. In addition to this, standard CCD instruments aboard a Star Tracker generate an image that is from 500-600 pixels in both width and height, for around 8X8 degree fields of view. The data from the DSS is shown with a pixel spacing of 530 pixels corresponding to 15 arcseconds, or 1 quarter of a degree, so obviously the DSS images are of much higher resolution than those of a normal star tracker. The images used for analysis will be of approximately 175X175 pixels (5X5 arcminutes), which is much smaller in terms of pixel count than the image generated by a star tracker, but should be large enough to validate the basic concept.

***Steps in the Algorithm*:**

Step 1: Once a star field array has been read into the MATHCAD program, it is necessary to determine what parts of the array will be treated as stars, and what parts of the array will be treated as space. For the purposes of this program, a "cut" is made, eliminating all array values less than 100, and setting them equal to zero. This value for the cut was selected based on trial and error, and a rough guess at how bright an object should be to be treated as a star. Whatever is left after the cut is bright enough to be considered a star. The output of this step is another matrix, similar to the original array read into the program, with elements less than 100 set equal to zero. This "cut," as implemented in MATHCAD, is shown as follows:

```
Zeroes(M) := | Cut ← 100
             | i ← 0
             | for m ∈ 0.. rows(M) − 1
             |   for n ∈ 0.. cols(M) − 1
             |     | $M_{m,n} \leftarrow 0$ if $M_{m,n} < Cut$
             |     | $L_{m,n} \leftarrow M_{m,n}$
             |     | i ← i + 1
             | L
```

Step 2: Once the original array has been cut to remove points not considered to be stars, it must be determined whether or not there are any points in the remaining array which might be considered to be stars. In order to do this, the numerical values of the array are evaluated at evenly space points. Those points which are greater than zero are probably a part of a star, and are set to a value of 1. Those points equal to zero are not part of a star, and are set to a value of 0. Once all of these points are evaluated, a True/False matrix is generated. The True/False matrix is the same size as the original array, and most of its elements will be set equal to zero. Those elements which are not equal to zero are parts of a star, although they may or may not be the actual center of the star. Implementation of this part of the algorithm is shown as follows:

```
TrueFalse(M) :=  Zeroes ← Zeroes(M)
                 for m ∈ 0.. rows(Zeroes) − 1
                   for n ∈ 0.. cols(Zeroes) − 1
                     $Zeroes_{m,n} \leftarrow 0$ if n=cols(M) − 1
                     $Zeroes_{m,n} \leftarrow 0$ if n=0
                     $Zeroes_{m,n} \leftarrow 0$ if m=0
                     $Zeroes_{m,n} \leftarrow 0$ if m=rows(M) − 1
                     $L_{m,n} \leftarrow 1$ if $Zeroes_{m,n} > 1$
                     $L_{m,n} \leftarrow 0$ if $Zeroes_{m,n} < 1$
                     $K_{m,n} \leftarrow L_{m,n}$ if $\frac{m}{3} = \mathrm{floor}\left(\frac{m}{3}\right)$
                     $K_{m,n} \leftarrow 0$ if
                         $\frac{m}{3} \neq \mathrm{floor}\left(\frac{m}{3}\right)$
                         $\frac{n}{3} \neq \mathrm{floor}\left(\frac{n}{3}\right)$
                 K
```

Step 3: One intermediate step useful for this program is an algorithm designed to record coordinates of a matrix whose values are greater than zero. This step of the program does this, and the output is list of array coordinates for each element of an array which is greater than zero. This is shown as follows:

```
Locate(M) :=  i ← 0
              L ← 0
              for m ∈ 0.. rows(M) − 1
                for n ∈ 0.. cols(M) − 1
                  if $M_{m,n} > 0$
                    $L^{<i>} \leftarrow \begin{bmatrix} m \\ n \end{bmatrix}$
                    i ← i + 1
              L
```

Step 4: In the True/False section of the program, it was determined which points within the array were points on a star. Once those points have been found, it is necessary to determine the coordinates of the actual center of the star. For the purposes of this algorithm, the center of the star will be considered to be the point of greatest numerical magnitude within the star. This section of the program solves for that point by generating a submatrix around each True/False point greater than zero. The coordinates of the center of the star will be treated as the coordinates of the point within the submatrix of greatest magnitude. This may not be the geometrical center of the star, but for the purposes of this paper, this will be the method used. The output of this section of the program consists of a series of coordinates. These coordinates are the possible coordinates for the center of each star. This part of the algorithm is as follows:

```
Bright(M) := | $L \leftarrow \text{Locate}(\text{TrueFalse}(M))$
             | $\text{TrueFalse} \leftarrow \text{TrueFalse}(M)$
             | $i \leftarrow 0$
             | while $i < \text{cols}(L)$
             |     | for $m \in 0..\text{rows}(M) - 1$
             |     |   for $n \in 0..\text{rows}(M) - 1$
             |     |     | $S \leftarrow \text{submatrix}\left[M, (L^{\langle i \rangle})_{0,0} - 2, (L^{\langle i \rangle})_{0,0} + 2, (L^{\langle i \rangle})_{1,0} - 2, (L^{\langle i \rangle})_{1,0} + 2\right]$
             |     |     | $K^{\langle i \rangle} \leftarrow \begin{bmatrix} m \\ n \end{bmatrix}$ if $\sqrt{\left[\left| m - (L^{\langle i \rangle})_{0,0} \right|\right]^2 + \left[\left| n - (L^{\langle i \rangle})_{1,0} \right|\right]^2} \le 5$ if $M_{m,n} = \max(S)$
             |     | $i \leftarrow i + 1$
             | $K$
```

Step 5: While the above portion of the algorithm solves for what are supposed to be the coordinates of the center of the star, there are many repeated identical coordinates, as well as many coordinates which are too close together to represent anything other than parts of one star. This next section of the algorithm eliminates identical coordinates, and also selects one coordinate set out of multiple coordinate sets that are probably a part of the same star.

```
LLocate (M) := | P ← Bright (M)
               | i ← 1
               | while  i < cols(P) − 1
               |     | $H^{<i>} \leftarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$  if  $\sqrt{(|P_{1,i} - P_{1,i-1}|)^2 + (|P_{0,i} - P_{0,i-1}|)^2} \le 5$
               |     | $H^{<i>} \leftarrow \begin{bmatrix} P_{0,i} \\ P_{1,i} \end{bmatrix}$  if  $\sqrt{(|P_{1,i} - P_{1,i-1}|)^2 + (|P_{0,i} - P_{0,i-1}|)^2} > 5$
               |     | i ← i + 1
               | V ← augment $(P^{<0>}, H)$
               | j ← 0
               | for  i ∈ 0.. cols(V) − 1
               |   if  $(V^{<i>})_{0,0} + (V^{<i>})_{1,0} \ne 0$
               |       | $C^{<j>} \leftarrow \begin{bmatrix} (V^{<i>})_{0,0} \\ (V^{<i>})_{1,0} \end{bmatrix}$
               |       | j ← j + 1
               | C
```

Step 6: Given the coordinates of all of the stars in the field of view, it is necessary to determine the coordinates of the star closest to the center of the matrix. This star will be the central star, and it is this star that will be identified in the program. In order to solve for the central star, the distances from all of the stars to the center of the matrix are calculated using the Pythagorean Theorem, and the star closest to the center of the matrix will be treated as the central star.

$$q(a,b,x,y,g) := \left| \sqrt{(a-x)^2 + (b-y)^2} \right| < g$$

```
starcenter(M, q) :=  | i ← 0
                     | L ← LLocate(M)
                     | Base ← 100
                     | for i ∈ 0.. cols(L) − 1
                     |   if q(L_{0,i}, L_{1,i}, floor(cols(M)/2), floor(rows(M)/2), Base) = 1
                     |     | S ← [L_{0,i} ; L_{1,i}]
                     |     | Base ← | sqrt((S_{0,0} − floor(cols(M)/2))^2 + (S_{1,0} − floor(rows(M)/2))^2) |
                     | S
```

Step 7: Now that a central star has been selected, the vectors from the central star to all of the stars in the field of view are calculated. The important part of this step is ensuring that all of the given vectors are in the correct quadrant in the x-y plane relative to the central star. Since the vectors are taken as the difference between the array coordinates of the central star and the star in question, it is necessary to make the x component of a vector positive if the first array coordinate of a star is greater than that of the central star, and negative if the first array coordinate of a star is less than that of the central star. Similarly, if the second array coordinate of a star is less than the second array component of the central star, the y component of the vector will be positive, and if the second array coordinate of a star is greater than the second array coordinate of the central star, the y component of the vector will be negative. The output of this part of the algorithm will be a series of vectors, from each star to the central star.

```
$\text{Vectors}(M) :=$
    $i \leftarrow 0$
    $L \leftarrow 0$
    $\text{Locate} \leftarrow \text{LLocate}(M)$
    $\text{starcenter} \leftarrow \text{starcenter}(M, q)$
    for $n \in 0 .. \text{cols}(\text{Locate}) - 1$
        $L^{\langle i \rangle} \leftarrow \begin{bmatrix} -(\text{starcenter}_{1,0} - \text{Locate}_{1,n}) \ \ \text{starcenter}_{1,0} > \text{Locate}_{1,n} \\ (\text{Locate}_{1,n} - \text{starcenter}_{1,0}) \ \text{if} \ \text{starcenter}_{1,0} \le \text{Locate}_{1,n} \\ (\text{starcenter}_{0,0} - \text{Locate}_{0,n}) \ \text{if} \ \text{starcenter}_{0,0} > \text{Locate}_{0,n} \\ (\text{Locate}_{0,n} - \text{starcenter}_{0,0}) \cdot -1 \ \text{if} \ \text{starcenter}_{0,0} \le \text{Locate}_{0,n} \end{bmatrix}$
        $i \leftarrow i + 1$
    $L$
```

Step 8: Since it is not needed to generate a data string, the origin vector (0,0) is pulled out of the list of vectors.

```
$\text{starvecs}(M) :=$
    $i \leftarrow 0$
    $L \leftarrow 0$
    $\text{Vectors} \leftarrow \text{Vectors}(M)$
    for $n \in 0 .. \text{cols}(\text{Vectors}) - 1$
        if $|\text{Vectors}_{0,n}| + |\text{Vectors}_{1,n}| \neq 0$
            $L^{\langle i \rangle} \leftarrow \begin{bmatrix} \text{Vectors}_{0,n} \\ \text{Vectors}_{1,n} \end{bmatrix}$
            $i \leftarrow i + 1$
    $L$
```

Step 9: As a preliminary step to finding angular displacements between stars, the angles from each star surrounding the central star are calculated, in an x-y plane. These angles are calculated as the arctangent of the y coordinate divided by the x coordinate of each vector, the sign of the angle being determined by the quadrant in which the vector lies. For those cases of a zero x value for the vector of a star, the angle is set to 90 or 270 degrees, to avoid a singularity. The output of this section of the algorithm is a series of angles, in no particular order.

```
Angles(M) :=
  i ← 0
  L ← 0
  starvecs ← starvecs(M)
  for n ∈ 0.. cols(starvecs) − 1
      L_i ← | if $\text{starvecs}_{0,n} = 0$
            |     $\frac{\pi}{2}$ if $\text{starvecs}_{1,n} > 0$
            |     $\frac{3\pi}{2}$ if $\text{starvecs}_{1,n} < 0$
            | if $\text{starvecs}_{0,n} \neq 0$
            |     if $\text{starvecs}_{0,n} > 0$
            |         $\text{atan}\left(\frac{\text{starvecs}_{1,n}}{\text{starvecs}_{0,n}}\right)$ if $\text{starvecs}_{1,n} > 0$
            |         $\text{atan}\left(\frac{\text{starvecs}_{1,n}}{\text{starvecs}_{0,n}}\right) + 2\cdot\pi$ if $\text{starvecs}_{1,n} < 0$
            |         $0$ if $\text{starvecs}_{1,n} = 0$
            |     if $\text{starvecs}_{0,n} < 0$
            |         $\text{atan}\left(\frac{\text{starvecs}_{1,n}}{\text{starvecs}_{0,n}}\right) + \pi$ if $\text{starvecs}_{1,n} > 0$
            |         $\text{atan}\left(\frac{\text{starvecs}_{1,n}}{\text{starvecs}_{0,n}}\right) + \pi$ if $\text{starvecs}_{1,n} < 0$
            |         $-\pi$ if $\text{starvecs}_{1,n} = 0$
      i ← i + 1
  L
```

Step 10: In this step, the angles in the x-y plane are sorted in order from least to greatest. The output is a series of angles, in order of least to greatest.

$$\text{Bestlist}(M) := \text{sort}(\text{Angles}(M))$$

Step 11: Because they are not needed, identical angles in the x-y plane are eliminated. This is done to allow for a check of the final data string, in which all of the angles between stars should add up to 360 degrees. This basically eliminates the angle for a star vector that is a linear multiple of another star vector, and while this does eliminate one star from the angular calculations, the impact on the desired data string should be somewhat negligible, since it would basically mean repeating an angle already calculated. The output is a series of angles, arranged in order from least to greatest, with no angle identical to another.

```
Sorter(M) := | $i \leftarrow 0$
             | $Bestlist \leftarrow Bestlist(M)$
             | $L_0 \leftarrow Bestlist_0$
             | $i \leftarrow 1$
             | for $n \in 1 .. rows(Bestlist) - 1$
             |   if $Bestlist_n \neq Bestlist_{n-1}$
             |       | $L_i \leftarrow Bestlist_n$
             |       | $i \leftarrow i + 1$
             | $L$
```

Step 12: In this step, the actual angles between the stars in the field of view are calculated. This is done by taking the differences between the angles in the x-y plane. The first angle calculated will be the difference between the smallest and largest angles in the x-y plane, subtracted from 360 degrees, as shown in figure 6:

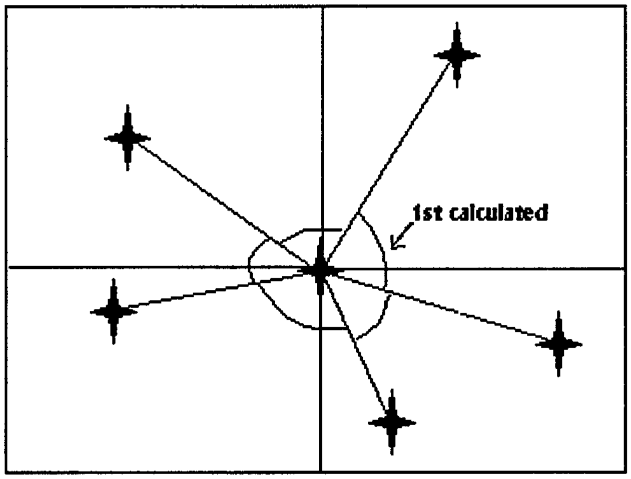


Figure 6: calculation of angles between stars

The rest of the angles will be calculated as the differences between angles. The output of this section of the algorithm is a series of angles, which add up to 360 degrees.

```
Reducesort(M) := | $i \leftarrow 0$
                 | $Sorter \leftarrow Sorter(M)$
                 | $k \leftarrow rows(Sorter) - 1$
                 | $L_0 \leftarrow Sorter_0 + (2 \cdot \pi - Sorter_k)$
                 | $i \leftarrow 1$
                 | for $n \in 1 .. rows(Sorter) - 1$
                 |     | $L_i \leftarrow Sorter_n - Sorter_{n-1}$
                 |     | $i \leftarrow i + 1$
                 | $L$
```

Step 13: In this step, the angles calculated above are ordered from least to greatest. The sort command in MATHCAD will sort the elements in an array in order from least to greatest. Output of this section of the algorithm will be a series of angles, in order from least to greatest. This series of angles is the data string, which will be compared to catalog data to determine a match.

$$\text{finalsort}(M) := \text{sort}(\text{Reducesort}(M))$$

Catalog Generation: For comparison purposes, a catalog was generated by augmenting the individual data strings for a group of stars and creating a catalog matrix. The columns of this catalog matrix represent the data strings for catalog stars. The augmentation process was done manually, and is shown as follows:

```
Cat1 := augment ( StringStar1 , StringStar2 )

Cat2 := augment ( Cat1 , StringStar3 )

Cat3 := augment ( Cat2 , StringStar4 )

Cat4 := augment ( Cat3 , StringStar6 )

Cat5 := augment ( Cat4 , StringStar7 )

Cat6 := augment ( Cat5 , StringStar8 )

Cat7 := augment ( Cat6 , StringStar9 )

Catalog := augment ( Cat7 , StringStar10 )
```

Step 14: At this point, the data string for a star can be compared to the catalog, in order to find a match. This comparison is done by comparing the elements in the data string for an observed star to the elements in the columns of the catalog matrix. When the sum of the differences between the elements of the catalog columns and the star data string is the smallest, the star corresponding to that column in the star catalog is identified as the star observed. The output of this section of code is the catalog star which most closely matches the observed star:

```
Match(M) := | StringM ← String(M)
            | Diff ← 100
            | for h ∈ 0.. cols(Catalog) − 1
            |     | for i ∈ 0.. rows(Catalog) − 1
            |     |     $Supp_i \leftarrow \left| \left(Catalog^{<h>}\right)_i - \left(StringM^{<0>}\right)_i \right|$
            |     | $Diffy_h \leftarrow \sum_{i=0}^{rows(Catalog)-1} Supp_i$
            |     | if $Diffy_h \le Diff$
            |     |     | $Diff \leftarrow Diffy_h$
            |     |     | Match ← h
            | Match
```

Results of Implementation

The algorithm and implementation discussed above was applied by first generating a catalog of 9 stars, and then comparing star fields to those nine stars, in hopes of finding a match. The catalog would have been larger, except for the fact that the fifth star field analyzed did not have enough stars to generate a catalog entry. Because of this, star field 5 was left out of the algorithm. The algorithm was first tested by comparing all of the catalog stars to the catalog. As expected, this test identified each tested catalog star as being a match to itself. After this comparison, an attempt was made to identify one of the catalog fields, after rotating it 90 degrees. This proved unsuccessful, and some reasons for this will be discussed in the next section. To check the calculation of the angles, a test was run on 10 by 10 test matrices, in which given patterns were rotated into different positions. The data strings for all of these tests were identical, indicating that the angular data is being properly calculated, and the weakness for application to an entire star field is probably a result of difficulties in distinguishing stars in the data arrays.

Discussion of Limitations

There are some definite weaknesses of the star identification algorithms presented here. One of the biggest involves the field of view used for cataloging and observation. In situations where observations are made using a small field of view, there will generally be only a few stars surrounding a central star which can be used for comparison. This means that the number of identifying features for a given star may be so small, that it will become difficult to distinguish the star from other stars in an equally small field of view. Smaller fields of view usually mean smaller numbers of identifying features, which will mean significant identification problems. On the other hand, using a very large field of view may generate such a large number of angles in the multi-star approach, that all of the stars begin to look alike, due to the very small angular separations between larger numbers of stars surrounding a central star. It is probable that there exists some intermediate field of view which will generated the best results for this algorithm, but the actual dimensions of an optimal field of view would probably require extensive testing.

For the time being, the angles calculated in the algorithm are organized from least to greatest, which weakens the quality of the search algorithm, because while the data string has been organized for comparison, the angular separations are no longer in the specific order found in observation. For instance, the angles between the stars in Figure 7 and Figure 8 are all identical, but they are clearly not the same stars.

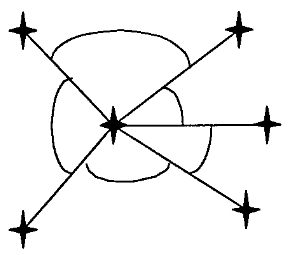


Figure 7.

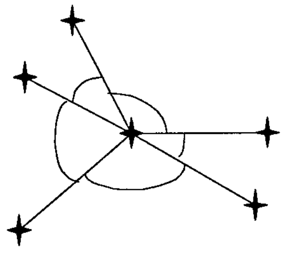


Figure 8.

The algorithm implemented here would not be able two discriminate between the central stars of Figures 7 and 8, because of the fact that it does not take into account the order of the angles between the nearby stars [2]. Despite the above noted problem, the program implemented does provide significant information for identification. It should be noted that there are hopefully not many star fields where the angles around the central star are all the same size, but appear in different order, especially when using small fields of view.

Unfortunately, one of the discoveries made researching this paper has been that the detailed algorithms and programming done to perform any basic identification technique are generally considered proprietary information, and are therefore protected by the aerospace companies doing this type of research and development. Both Ball Aerospace and Lockheed Martin have star trackers, and the accompanying identification algorithms and software, but were unwilling to make that information available because of the competitive nature of the market for these items.

Possibilities For Improvement

There are a variety of ways to improve the quality of the algorithm presented here. One way which is gaining interest involves the use of a neural network, in which a catalog of stars is divided up into various classes [6]. When a star is observed, it is then determined which class of stars the observed star most likely falls into. The star is then compared to all of the stars in that specific class, in hopes of finding a likely match. One quick way of determining how to classify an observed star might be the use of a non-geometrical characteristic, such as magnitude, or spectral type. Magnitude classification would probably be easiest to implement right now, because CCD observations can determine the magnitude of an observed star quite easily. While CCD observations can take a certain amount of spectral data, determining the spectral type of a star might take either a larger number of CCDs, or the addition of another instrument, such as a spectrometer, to a star tracking device. This latter option will probably not be very popular, due to the increase in weight and complexity.

Discussion of Applications

The importance of star tracking is obvious. In the space environment, the ability to rapidly orient a spacecraft can be very important, and rapidly determining attitude is a significant step in accomplishing this task. Rapid star identification contributes to rapid attitude determination and control, so developing quick, robust identification algorithm is an important task. Missions with high pointing accuracy requirements will continue to require improved identification techniques.

One example of a mission with a need for a star tracker is an interplanetary mission. There is a star imaging unit aboard the Cassini spacecraft, and it is used for attitude determination purposes [1]. The advantage of using a star imaging attitude sensor for interplanetary missions is that the positions of the stars will remain essential fixed relative to an interplanetary probe for the duration of a mission. Other methods of attitude determination used for Earth orbiting missions, such as magnetometers, sun sensors, or horizon sensors could not be used for interplanetary missions, because there is no horizon or magnetic field to measure. In these situations, star trackers are superior, because the star reference frame is constant for mission duration.

Another possible application for this type of technology is in planetary navigation. A zenith pointed star sensor could be used to determine the point in the celestial sphere which is directly above a planetary rover. When combined with knowledge of the time of observation, it would be possible to determine position with some degree of accuracy [9].

More common applications of this type of technology are oriented towards rapid attitude acquisition, improving other attitude measurements, and fast recovery of attitude knowledge [9].

Conclusions

The algorithm designed and implemented as part of this paper was moderately successful at identifying stars. Obviously, the implementation of the program was not done with a large enough catalog to justify making any concrete conclusions as to the value of this particular algorithm. Also, it may have been a good idea to have implemented one of the other approaches to the problem, such as the tri-star approach, in hopes of comparing the tri-star concept with the multi-star concept developed here. This paper has, however, validated the basic idea of the multi-star concept, and the implementation can probably be further refined to make it more effective.

Further work on this type of algorithm would be dedicated towards improving the speed at which an identification was achieved, and also attempting to lessen the number of identification runs which generate false identifications. Overall, the project has proved some basic ideas with regards to star identification, and the need for further research in this area is evident.

Bibliography

1. Alexander, J.W., and Daniel H. Chang. "Scene Simulation for Real-Time testing of the Cassini Star Tracking and Identification functions." AAS paper, February, 1995.

2. Augusteijn, Marijke. Personal Interview. April, 1998

3. Eterno, J.S., Zermuehlen, R.O., and Harold F. Zimbelman "Attitude Determination and Control", Space Mission Analysis and Design. Torrence: Kluwer Academic Publishers, 1997.

4. Gottlieb, David M. "Star Identification Techniques," Spacecraft Attitude Determination and Control, Boston MA: D. Reidel, 1978.

5. Ketchum, E. A., and Robert H. Tolson, "Onboard Star Identification Without A Priori Attitude Information," Journal of Guidance, Control, and Dynamics, Vol 18, No 2, March-April 1995.

6. Lindsey, C. S., Lindblad, T, and Age Eide. "A Method for Star Identification Using Neural Networks," SPIE Vol.3077 0277-786X/97.

7. Scholl, M.S. "Star Field Identification for Autonomous Attitude Determination," Journal of Guidance, Control, and Dynamics, Vol 18, No.1, January-February, 1995.

8. The Digitized Sky Surveys were produced at the Space Telescope Science Institute under U.S. Government grant NAG W-2166. The images of these surveys are based on photographic data obtained using the Oschin Schmidt Telescope on Palomar Mountain and the UK Schmidt Telescope. The plates were processed into the present compressed digital form with the permission of these institutions.

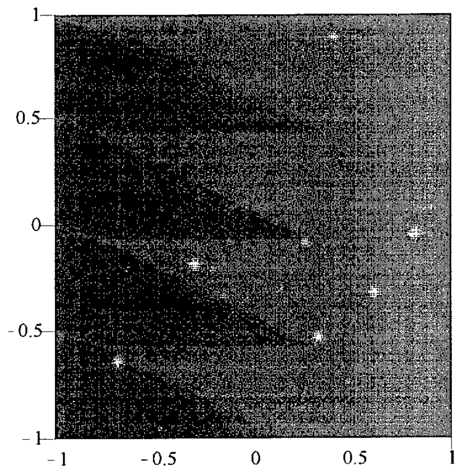
9. Van Bezooijen, R.W. "Autonomous Star Trackers for Geostationary Satellites," SPIE paper, Aug. 1996.

10. Woodard, M. and Dave Rohrbaugh, "Development of a Robust Star Identification Technique for Use in Attitude Determination of the ACE Spacecraft," NASA report, Jan. 1998, Online at: http://fdd.gsfc.nasa.gov/mwoodard/ACE/FMET_paper.html .

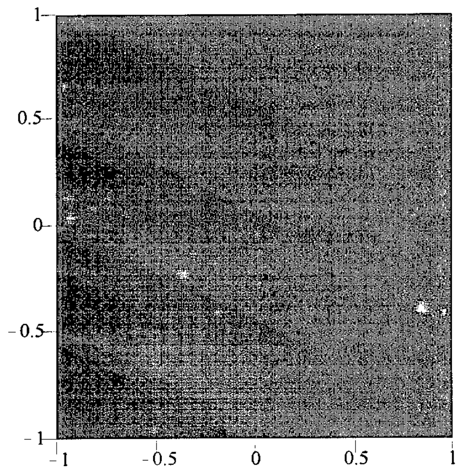
Appendix A

As a preliminary step to the algorithm, a series of pictures has been placed into the program, for analysis. These pictures are input and displayed as follows:

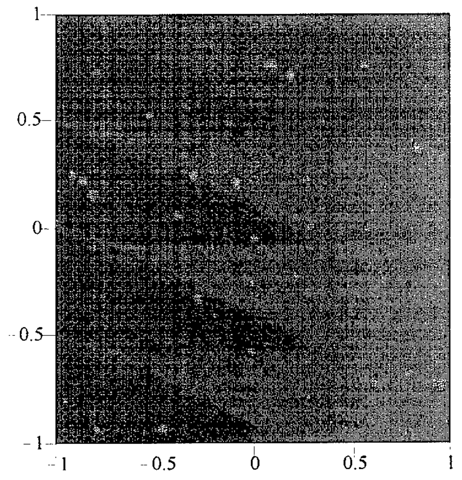
```
Star1  = READ_IMAGE("A:/star1.gif")        Star2  = READ_IMAGE("A:/star2.gif")

Star3  = READ_IMAGE("A:/star3.gif")        Star4  = READ_IMAGE("A:/star4.gif")

Star5  = READ_IMAGE("A:/star5.gif")        Star6  = READ_IMAGE("A:/star6.gif")

Star7  = READ_IMAGE("A:/star7.gif")        Star8  = READ_IMAGE("A:/star8.gif")

Star9  = READ_IMAGE("A:/star9.gif")        Star10 = READ_IMAGE("A:/star10.gif")
```



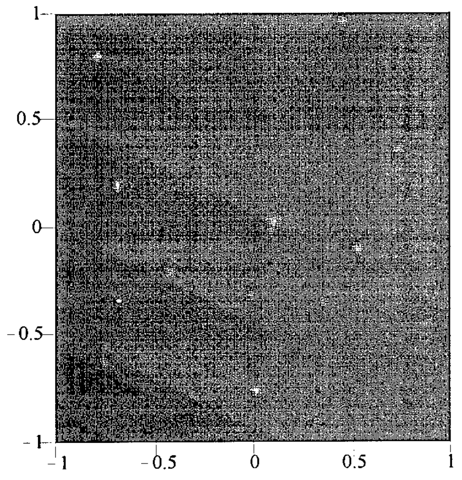
Star1



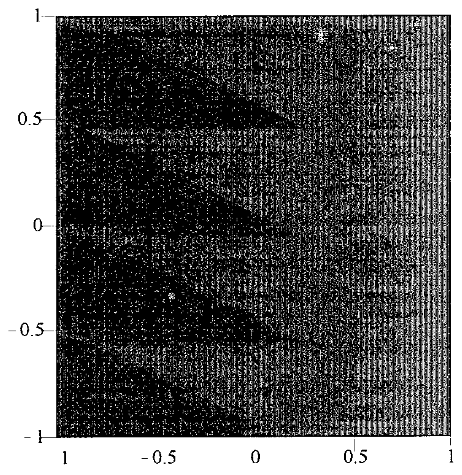
Star2



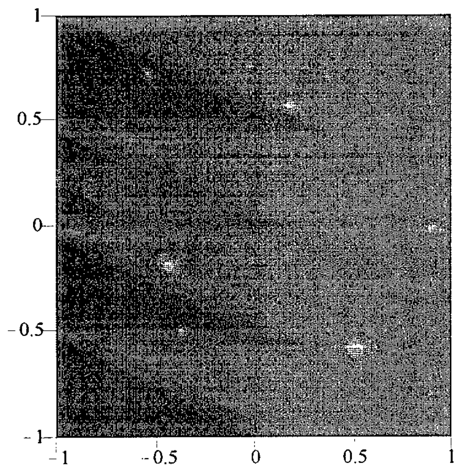
Star3



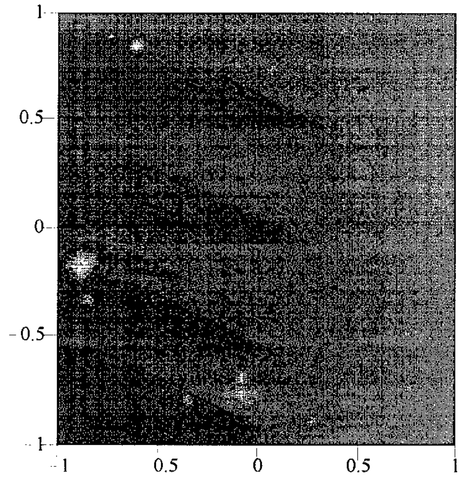
Star4

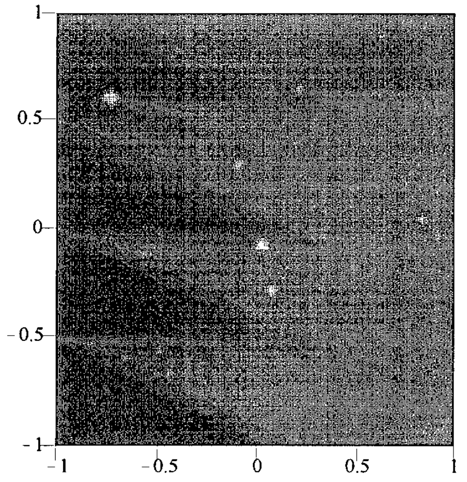


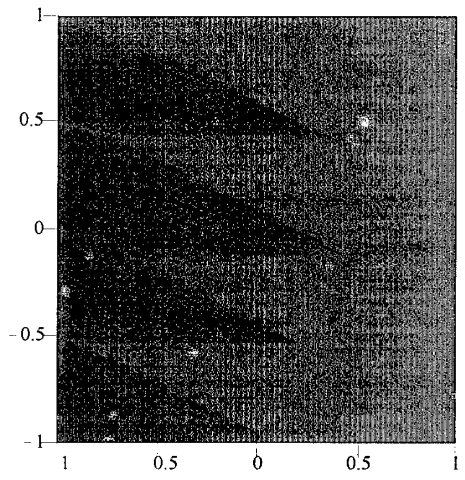
Star5

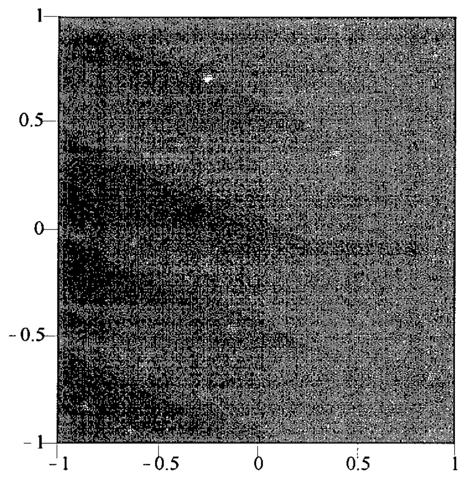


Star6


Star7


Star8


Star9


Star10

One of the first steps in the star identification process outlined below is to process the actual image. The process outlined below is somewhat simple, but does do basic processing for the purposes of this project.

One of the first steps involved in the image processing is deciding how bright an object in the observation field must be to be called a star. For the purposes of this paper, the cuttoff point will be magnitude of less than 100 registered at an individual pixel. Anything of magnitude less than 100 will be set to zero.

```
Zeroes(M) := | Cut ← 100
             | i ← 0
             | for m ∈ 0..rows(M) − 1
             |   for n ∈ 0..cols(M) − 1
             |     | $M_{m,n} \leftarrow 0$ if $M_{m,n} < Cut$
             |     | $L_{m,n} \leftarrow M_{m,n}$
             |     | i ← i + 1
             | L
```

Once the pixels of magnitude less than zero have been set equal to zero, it is necessary to determine whether or not there are any possible star candidates in a given observational field. In order to do this, I have chose to treat each star as a blob. The True/False algorithm determines for equally spaced points in the field whether or not there is an actual star near the point. The true condition is set if the pixel in question has a magnitude greater than 100, and is hence on part of the "blob" representing a star. The False condition is set for situations in which a pixel is not touching a part of the blob. The output of this algorithm is a Matrix of the same size as the star field in question, with ones within a the matrix representing points close to the actual location of a star.

```
TrueFalse(M) := | Zeroes ← Zeroes(M)
                | for m ∈ 0.. rows(Zeroes) − 1
                |   for n ∈ 0.. cols(Zeroes) − 1
                |     | Zeroes_{m,n} ← 0  if n=cols(M) − 1
                |     | Zeroes_{m,n} ← 0  if n=0
                |     | Zeroes_{m,n} ← 0  if m=0
                |     | Zeroes_{m,n} ← 0  if m=rows(M) − 1
                |     | L_{m,n} ← 1  if Zeroes_{m,n} > 1
                |     | L_{m,n} ← 0  if Zeroes_{m,n} < 1
                |     | K_{m,n} ← L_{m,n}  if $\frac{m}{3}$=floor($\frac{m}{3}$)
                |     | K_{m,n} ← 0  if | $\frac{m}{3}$≠floor($\frac{m}{3}$)
                |     |                 | $\frac{n}{3}$≠floor($\frac{n}{3}$)
                | K
```

The coordinate location function below is used to determine the coordinates of specific points within a matrix which meet a certain criteria. For this specific example, the criteria is being greater than zero. This function will be important in determining the actual coordinates of a star of a star. the output is a series of coordinates, representing points within the matrix which meet the aforementioned criteria.

```
Locate(M) := | $i \leftarrow 0$
             | $L \leftarrow 0$
             | for $m \in 0 .. \text{rows}(M) - 1$
             |   for $n \in 0 .. \text{cols}(M) - 1$
             |     if $M_{m,n} > 0$
             |       | $L^{\langle i \rangle} \leftarrow \begin{bmatrix} m \\ n \end{bmatrix}$
             |       | $i \leftarrow i + 1$
             | $L$
```

The above function has successfully found a point near the point of greatest Magnitude of the star. Now it is necessary to identify the actual center of the star, which for the purposes of this paper will be the point of greatest magnitude within the "blob" which represents the star. The Bright algorithm is designed to generate a submatrix of the original star field around the points in the True/False matrix which register true. The m,n coordinates of the point within this submatrix that is the maximum value of all points within the submatrix are then treated as the actual m,n coordinates of the star.

```
Bright(M) := | $L \leftarrow Locate(TrueFalse(M))$
             | $TrueFalse \leftarrow TrueFalse(M)$
             | $i \leftarrow 0$
             | while $i < cols(L)$
             |    | for $m \in 0..rows(M) - 1$
             |    |   for $n \in 0..rows(M) - 1$
             |    |      | $S \leftarrow submatrix\left[M, \left(L^{<i>}\right)_{0,0} - 2, \left(L^{<i>}\right)_{0,0} + 2, \left(L^{<i>}\right)_{1,0} - 2, \left(L^{<i>}\right)_{1,0} + 2\right]$
             |    |      | $K^{<i>} \leftarrow \begin{bmatrix} m \\ n \end{bmatrix}$ if $\sqrt{\left[\left| m - \left(L^{<i>}\right)_{0,0} \right|\right]^2 + \left[\left| n - \left(L^{<i>}\right)_{1,0} \right|\right]^2} \le 5$ if $M_{m,n} = max(S)$
             |    | $i \leftarrow i + 1$
             | K
```

This section of programming statements further resolves the coordinates solved for in the Bright algorithm by eliminating identical coordinates (a problem which occurs for multiple True registers in the True/False matrix for the same star), and eliminating all but one of any series of coordinates too close together to represent anything more than a single star

```
$\mathrm{LLocate}(M) :=$
    $P \leftarrow \mathrm{Bright}(M)$
    $i \leftarrow 1$
    while $i < \mathrm{cols}(P) - 1$
        $H^{\langle i \rangle} \leftarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ if $\sqrt{\left(\left| P_{1,i} - P_{1,i-1} \right|\right)^2 + \left(\left| P_{0,i} - P_{0,i-1} \right|\right)^2} \le 5$
        $H^{\langle i \rangle} \leftarrow \begin{bmatrix} P_{0,i} \\ P_{1,i} \end{bmatrix}$ if $\sqrt{\left(\left| P_{1,i} - P_{1,i-1} \right|\right)^2 + \left(\left| P_{0,i} - P_{0,i-1} \right|\right)^2} > 5$
        $i \leftarrow i + 1$
    $V \leftarrow \mathrm{augment}\left(P^{\langle 0 \rangle}, H\right)$
    $j \leftarrow 0$
    for $i \in 0 .. \mathrm{cols}(V) - 1$
        if $\left(V^{\langle i \rangle}\right)_{0,0} + \left(V^{\langle i \rangle}\right)_{1,0} \ne 0$
            $C^{\langle j \rangle} \leftarrow \begin{bmatrix} \left(V^{\langle i \rangle}\right)_{0,0} \\ \left(V^{\langle i \rangle}\right)_{1,0} \end{bmatrix}$
            $j \leftarrow j + 1$
    $C$
```

Once the LLocate algorithm has solved for the coordinates of all of the stars in a given field of view, it is necessary to determine the center star for all of the stars found in the matrix. This center star is the star which will it is hoped will be identified.

$$q(a,b,x,y,g) := \left|\sqrt{(a-x)^2+(b-y)^2}\right| < g$$

```
starcenter(M, q) :=
    $i \leftarrow 0$
    $L \leftarrow \text{LLocate}(M)$
    $\text{Base} \leftarrow 100$
    for $i \in 0 .. \text{cols}(L) - 1$
        if $q\left(L_{0,i}, L_{1,i}, \text{floor}\left(\frac{\text{cols}(M)}{2}\right), \text{floor}\left(\frac{\text{rows}(M)}{2}\right), \text{Base}\right) = 1$
            $S \leftarrow \begin{bmatrix} L_{0,i} \\ L_{1,i} \end{bmatrix}$
            $\text{Base} \leftarrow \left|\sqrt{\left(S_{0,0} - \text{floor}\left(\frac{\text{cols}(M)}{2}\right)\right)^2 + \left(S_{1,0} - \text{floor}\left(\frac{\text{rows}(M)}{2}\right)\right)^2}\right|$
    $S$
```

The Vectors Algorithm solves for the vectors of the stars surrounding the central star in a rectancular coordinate system, with the center star within the matrix being treated as the origin.

```
$\mathrm{Vectors}(M) :=$
    $i \leftarrow 0$
    $L \leftarrow 0$
    $\mathrm{Locate} \leftarrow \mathrm{LLocate}(M)$
    $\mathrm{starcenter} \leftarrow \mathrm{starcenter}(M, q)$
    for $n \in 0 .. \mathrm{cols}(\mathrm{Locate}) - 1$
        $L^{\langle i \rangle} \leftarrow \begin{bmatrix} \left\| \begin{array}{l} -(\mathrm{starcenter}_{1,0} - \mathrm{Locate}_{1,n})\ \mathrm{starcenter}_{1,0} > \mathrm{Locate}_{1,n} \\ (\mathrm{Locate}_{1,n} - \mathrm{starcenter}_{1,0})\ \text{ if }\ \mathrm{starcenter}_{1,0} \le \mathrm{Locate}_{1,n} \end{array} \right. \\ \left\| \begin{array}{l} (\mathrm{starcenter}_{0,0} - \mathrm{Locate}_{0,n})\ \text{ if }\ \mathrm{starcenter}_{0,0} > \mathrm{Locate}_{0,n} \\ (\mathrm{Locate}_{0,n} - \mathrm{starcenter}_{0,0}) \cdot -1\ \text{ if }\ \mathrm{starcenter}_{0,0} \le \mathrm{Locate}_{0,n} \end{array} \right. \end{bmatrix}$
        $i \leftarrow i + 1$
    $L$
```

The vectors above have been solved for, but now it is necessary to pull the origin vector (0,0) out of the vectors. The algorithm starvecs does this:

```
$\mathrm{starvecs}(M) :=$
    $i \leftarrow 0$
    $L \leftarrow 0$
    $\mathrm{Vectors} \leftarrow \mathrm{Vectors}(M)$
    for $n \in 0 .. \mathrm{cols}(\mathrm{Vectors}) - 1$
      if $|\mathrm{Vectors}_{0,n}| + |\mathrm{Vectors}_{1,n}| \ne 0$
        $L^{\langle i \rangle} \leftarrow \begin{bmatrix} \mathrm{Vectors}_{0,n} \\ \mathrm{Vectors}_{1,n} \end{bmatrix}$
        $i \leftarrow i + 1$
    $L$
```

Once the vectors of the surrounding stars to the central star have been calculated, it is necessary to determine what the angles (measured in radians), of each surrounding star are, as measured in the previously mentioned x-y plane. The algorithm Angles does this, by implementing Euclidian Geometry:

```
Angles(M) := i ← 0
             L ← 0
             starvecs ← starvecs(M)
             for n ∈ 0.. cols(starvecs) − 1
                 L_i ← if starvecs_{0,n} = 0
                           $\frac{\pi}{2}$  if starvecs_{1,n} > 0
                           $\frac{3\pi}{2}$  if starvecs_{1,n} < 0
                       if starvecs_{0,n} ≠ 0
                           if starvecs_{0,n} > 0
                               $\operatorname{atan}\left(\frac{starvecs_{1,n}}{starvecs_{0,n}}\right)$  if starvecs_{1,n} > 0
                               $\operatorname{atan}\left(\frac{starvecs_{1,n}}{starvecs_{0,n}}\right) + 2\cdot\pi$  if starvecs_{1,n} < 0
                               0  if starvecs_{1,n} = 0
                           if starvecs_{0,n} < 0
                               $\operatorname{atan}\left(\frac{starvecs_{1,n}}{starvecs_{0,n}}\right) + \pi$  if starvecs_{1,n} > 0
                               $\operatorname{atan}\left(\frac{starvecs_{1,n}}{starvecs_{0,n}}\right) + \pi$  if starvecs_{1,n} < 0
                               $-\pi$  if starvecs_{1,n} = 0
                 i ← i + 1
             L
```

The above section of code accurately calculates the x-y angles of all stars in the Matrix!

It is now necessary to sort the angles calculated above in order from least to greatest. This is the smallest line in the program

$$\text{Bestlist}(M) := \text{sort}(\text{Angles}(M))$$

Now the angles calculated above can be calculated into the more general angles between the stars near the star of interest. As a first step, however, we eliminate any angles in the x-y plane system that are identical. This is only necessary for situations in which one of the star vectors is an integer multiple of another star vector. For instance in a situation where one star vector is (1,1), and another star vector is (2,2), only one angle will be used for the two stars.

```
Sorter(M) :=
    $i \leftarrow 0$
    $\text{Bestlist} \leftarrow \text{Bestlist}(M)$
    $L_0 \leftarrow \text{Bestlist}_0$
    $i \leftarrow 1$
    for $n \in 1 .. \text{rows}(\text{Bestlist}) - 1$
      if $\text{Bestlist}_n \neq \text{Bestlist}_{n-1}$
          $L_i \leftarrow \text{Bestlist}_n$
          $i \leftarrow i + 1$
    $L$
```

Now that any uneccesary angles have been eliminated, the actual angles between the stars are calculated in the Reducesort algorithm:

```
Reducesort(M) :=  i ← 0
                  Sorter ← Sorter(M)
                  k ← rows(Sorter) − 1
                  $L_0 \leftarrow Sorter_0 + (2\cdot\pi - Sorter_k)$
                  i ← 1
                  for n ∈ 1 .. rows(Sorter) − 1
                      $L_i \leftarrow Sorter_n - Sorter_{n-1}$
                      i ← i + 1
                  L
```

finalsort(M) := sort(Reducesort(M))

Now that the above angles have been reduced to the proper order, they can be put into an identification string. this identification string will be used for comparing the data in the first observation to a catalog of data sets.

```
String(M) :=  reduce ← finalsort(M)
              for n ∈ 0 .. rows(reduce) − 1
                  $Str_n \leftarrow reduce_n$
              for m ∈ rows(reduce) .. 10
                  $Str_m \leftarrow 0$
              Str
```

In order to determine which star an observed star is, it will be necessary to have a catalog for comparison purposes. The following nine data strings will be used as a catalog for nine different stars, which the central star in an observed field of view will be compared to: (Notice that star 5 does not have a data string, since it lacked a good central star.

$$\mathrm{StringStar1} := \begin{bmatrix} 0.361 \\ 0.638 \\ 0.851 \\ 1.153 \\ 1.377 \\ 1.904 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathrm{StringStar2} := \begin{bmatrix} 0.016 \\ 0.113 \\ 0.414 \\ 0.418 \\ 1.87 \\ 3.452 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathrm{StringStar3} := \begin{bmatrix} 0.195 \\ 0.353 \\ 0.478 \\ 0.53 \\ 0.8 \\ 1.878 \\ 2.049 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathrm{StringStar4} := \begin{bmatrix} 0.225 \\ 0.227 \\ 0.339 \\ 0.363 \\ 0.5 \\ 0.584 \\ 0.631 \\ 0.669 \\ 1.211 \\ 1.534 \\ 0 \end{bmatrix} \quad \mathrm{StringStar6} := \begin{bmatrix} 0.041 \\ 0.522 \\ 0.701 \\ 0.897 \\ 4.122 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathrm{StringStar7} := \begin{bmatrix} 0.307 \\ 0.673 \\ 0.93 \\ 1.07 \\ 3.303 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{StringStar8} := \begin{bmatrix} 0.529 \\ 0.54 \\ 2.545 \\ 2.668 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{StringStar9} := \begin{bmatrix} 0.014 \\ 0.084 \\ 0.783 \\ 1.929 \\ 3.473 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{StringStar10} := \begin{bmatrix} 0.14 \\ 0.536 \\ 0.791 \\ 1.142 \\ 3.674 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Cat1 := augment(StringStar1 , StringStar2)

Cat2 := augment(Cat1 , StringStar3)

Cat3 := augment(Cat2, StringStar4)

Cat4 := augment(Cat3 , StringStar6)

Cat5 := augment(Cat4, StringStar7)

Cat6 := augment(Cat5, StringStar8)

Cat7 := augment(Cat6, StringStar9)

Catalog := augment(Cat7, StringStar10)

The commands to the left are joining the individual data strings for each star into a Catalog matrix, with each column of the matrix representing a specific star.

The Catalog Matrix is shown as follows:

Catalog =

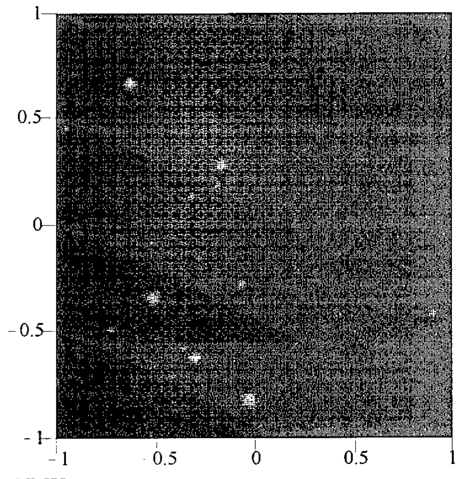
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	6	7	8
0.361	0.016	0.195	0.225	0.041	0.307	0.529	0.014	0.14
0.638	0.113	0.353	0.227	0.522	0.673	0.54	0.084	0.536
0.851	0.414	0.478	0.339	0.701	0.93	2.545	0.783	0.791
1.153	0.418	0.53	0.363	0.897	1.07	2.668	1.929	1.142
1.377	1.87	0.8	0.5	4.122	3.303	0	3.473	3.674
1.904	3.452	1.878	0.584	0	0	0	0	0
0	0	2.049	0.631	0	0	0	0	0
0	0	0	0.669	0	0	0	0	0
0	0	0	1.211	0	0	0	0	0
0	0	0	1.534	0	0	0	0	0
0	0	0	0	0	0	0	0	0

The following section of code is where it is determined which star of the stars in the catalog an observed star is the most similar to. This is done by summing up the differences between all of the angles, and then setting the star catalog with the smallest sum of differences as being the actual star:

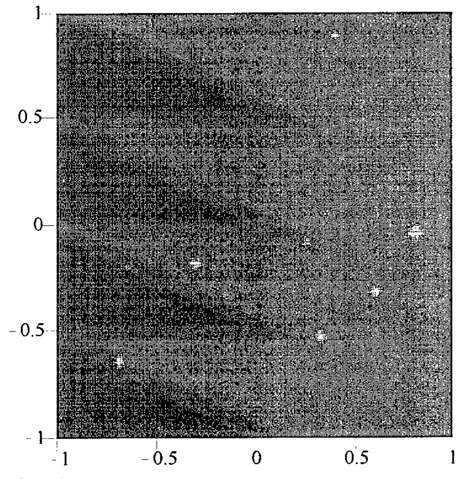
```
Match(M) :=  StringM ← String(M)
             Diff ← 100
             for h ∈ 0.. cols(Catalog) − 1
                 for i ∈ 0.. rows(Catalog) − 1
                     $Supp_i \leftarrow \left| \left(Catalog^{\langle h \rangle}\right)_i - \left(StringM^{\langle 0 \rangle}\right)_i \right|$
                 $Diffy_h \leftarrow \sum_{i=0}^{rows(Catalog)-1} Supp_i$
                 if $Diffy_h \le Diff$
                     $Diff \leftarrow Diffy_h$
                     Match ← h
             Match
```

The above Matching algorithm matched correctly identified all of the catalog stars, based on the pictures used to generate the catalog. To further test the algorithm, the picture used for the first catalog star was rotated 90 degrees, and run through the Algorithm:

```
UNK := READ_IMAGE( "A:/starunk.gif" )
```



UNK



Star1

When the "unknown" star was run through the matching algorithm, it was not properly identified. A probable reason for this is that the star imaging process was not working correctly, but it would require further development to determine the source of this problem.

To test the basic concept of generating the data strings, a series of 10X10 matrices were generated, and then run through the algorithm. The matrices all represent the same star field, in terms of angular separations, but have been rotated in various ways for testing. Two of these are shown. The data strings for both are identical, indicating that the angle determination and matching technique works, and the problem with the algorithm is probably in initial processing.

$$\mathrm{Mat1} := \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&1&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \qquad \mathrm{Mat2} := \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&1&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\mathrm{String}(\mathrm{Mat1}) = \begin{array}{|l} 0.588 \\ 0.588 \\ 2.554 \\ 2.554 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \qquad \mathrm{String}(\mathrm{Mat2}) = \begin{array}{|l} 0.588 \\ 0.588 \\ 2.554 \\ 2.554 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}$$